Application/Control Number: 10/709,501

Attention: Todd Ingberg

Inventor: Igor Voln

Correspondence Address:
2464 Prince Edward St Apt 1017
Honolulu HI 96815

Phone: (510) 316-1268
Email: idht@yahoo.com

Inventor's statement: I am a pro se applicant. I am requesting constructive assistance.

According to our conversation I went through your letter (Art Unit 2193). Please find my reply below. A soft copy of all documents is on the enclosed CD. Feel free to contact me if something is not clear. I am in Hawaii (standard time zone: UTC/GMT -10 hours)

To make my answer clearer I will be going through the numbered paragraphs in the DETAILED ACTION section of your letter.

1. No action required
2. Drawings for Figures 2, 3, 5, 8, 9, and 12 are enclosed on paper and CD. Shading is corrected.
3. Substitute specification because of font and headers: abstract is too long. The corrected files are enclosed.
4. Claims 1 and 5: periods removed, modified according to our conversation. The Word document with "Track Changes" set to "on" is provided on paper and CD. The final version is available in Word and PDF (see files on the CD)

5. Claim 13 about source control has been slightly changed and explained below. Claim 15 about commonly accepted code indentation rules has been changed and explained below.
6. No action required
7. The invented *method* takes some content as input, applies the *user defined* rules, and builds a tree-like representation. The result may or may not be the concrete hard-coded tree-like structure that James Martin describes.

The first misunderstanding that I would like to clarify is that the word "method" in claims refers to a process, not to the final output of the process. The invented method takes some content as input, applies the user defined rules, and builds a tree-like representation.

The rules are defined by the user according to the user needs at the moment. There could be a few different trees for the same content. Each tree could be created using a different rule set. For example, one of the trees may be built according to the rules that James describes. The other one only displays the “if” statements and does not show anything else. Yet another tree only shows method names. Yet other tree shows only comments. One more shows nodes filtered using some criteria. Yet another substitutes some parts of the document with user selected tags or controls. The output of the rules is not necessarily a set of tree nodes. The mapping could command to present a specific node as a diagram, a picture, a clickable interactive control etc. That is why the term “tree like” is used instead of simply the “tree”. James Martin says nothing about presentation of individual lines on the resulting tree. He simply displays them as *strings*. He does not introduce any auxiliary tokens that are not present in the original content, like the token “record” displayed on Fig. 2. Likewise, he is not talking about placing “if” and “else” on different levels like displayed on Fig. 14. A more useful example that I am planning to implement in the future is adding the performance view, which would display estimated performance data for each statement either as a sub-node in the tree, or as an in-tree control. Meaning, the mapping rules have the option to look up or calculate more data than available from the content itself. Another option would be displaying the assembler commands that a line or token would generate, which is still included in the term "representation", but this representation is essentially some symbols or controls. Yet another representation user could create could be presenting VB.NET code as a VC# tree. This conversion should be possible, because user has full control over mapping rules and can simply replace the positioning of tokens in mapping. I assume that the reader understands that the representation grammar is based on the structured content grammar, but in a general case it is different. I think, it should all be covered by Claim 9, let me know if you think I need to rephrase it.

It is clear that the system that builds **any** representation meets the user needs much better than the single hard-coded representation that James Martin described in his book. His presentation is only a particular case of the output of the claimed system.

The tree form that James describes is hard-coded based on what the author **thinks** is a convenient representation. There could be many real life situations, when the user would want to see a differently build tree. For example, I could only be interested in the "if" statements and have no desire to see anything else. I change the rules appropriately and can see the outline of the document only in terms of the "if" statements that the document contains. What is important, the output could still show original content tokens or something instead of them. The flexibility of presenting elements in a particular way could make a significant difference in terms of node manipulations and program development, user could frequently manipulate only a subset of the record instead of the whole record and might want to create a parent-child relationship for a particular set of lines only even though these lines are NOT linked by "begin-end" type of relationship that James Martin puts as a basis of his tree.

Claim 13: Source control related claim suggests that one of the representation trees could be used for source control purposes. The tree does not modify the original content but subdivides it into sections each of which could be individually locked by a user for editing. It is a definite progress because if a user locks the file as a whole, nobody else is able to work with it. If the user is able to lock only a node of representation tree, other users could edit other parts of the same content at the same time.

Claim 15: If content is messed up it is hard to understand how it works and modify it. For example, JavaScript files available on the web are often shrunk to provide for smaller size and quicker download. Spaces, tabs, and CR-LFs are removed etc. A representation tree that is built according to commonly used indentation rules would still present the content in an easy to read fashion. This is what the claim 15 is about. Let me know if you think the updated version covers it clearly.

Note: I have set up a web site for a product based on the ideas described in this patent. Unfortunately, I do not have financing at this time and am restricted in terms of resources, so I could not implement every feature described in the patent yet. However, feel free to check it out if you think it will be useful for your work. The URL is:

http://www.officialsourceexplorer.com

The mapping rules editor is not available in the downloadable version; however, there is a screenshot in the online documentation.